原理理解

1. **01 python的学习** 完成了python语言及其关键库Numpy，Pytorch的学习。为后续模型开发奠定了技术基础。
    - 学习材料是什么？在复现中、近期科研中，基础技能做了什么贡献？
        - **学习材料：**
        - **看懂并知道如何改进框架的代码**，比如说，我们知道处理视频数据的输入输出应该是什么，知道如何构建模型架构、如何加载数据之类，还能通过优化代码逻辑和调用 GPU，实现模型的高效训练。

```
def forward(self, x):  # Batch_size*[3, T, 128,128]
    x_visual = x
    [batch, channel, length, width, height] = x.shape

    x = self.ConvBlock1(x)  # x [3, T, 128,128]
    x = self.MaxpoolSpa(x)  # x [16, T, 64,64]

    x = self.ConvBlock2(x)  # x [32, T, 64,64]
    x_visual6464 = self.ConvBlock3(x)  # x [32, T, 64,64]
    # x [32, T/2, 32,32]    Temporal halve
    x = self.MaxpoolSpaTem(x_visual6464)

    x = self.ConvBlock4(x)  # x [64, T/2, 32,32]
    x_visual3232 = self.ConvBlock5(x)  # x [64, T/2, 32,32]
    x = self.MaxpoolSpaTem(x_visual3232)  # x [64, T/4, 16,16]

    x = self.ConvBlock6(x)  # x [64, T/4, 16,16]
    x_visual1616 = self.ConvBlock7(x)  # x [64, T/4, 16,16]
    x = self.MaxpoolSpa(x_visual1616)  # x [64, T/4, 8,8]

    x = self.ConvBlock8(x)  # x [64, T/4, 8, 8]
    x = self.ConvBlock9(x)  # x [64, T/4, 8, 8]
    x = self.upsample(x)  # x [64, T/2, 8, 8]
    x = self.upsample2(x)  # x [64, T, 8, 8]

    # x [64, T, 1,1]    -->  groundtruth left and right - 7
    x = self.poolspa(x)
    x = self.ConvBlock10(x)  # x [1, T, 1,1]

    rPPG = x.view(-1, length)

    return rPPG, x_visual, x_visual3232, x_visual1616
```

```
def forward(self, x):
    # ### 1. 输入层 ###
    # 输入 x 的维度通常为 [Batch, 3, T, 128, 128]
    # 含义：[批大小, RGB通道, 时间帧数, 高, 宽]
    # 这里是原始视频帧输入
    x_visual = x

    # 获取输入维度的具体数值，方便后续做 view 操作
    [batch, channel, length, width, height] = x.shape

    # ### 2. 浅层特征提取 (Encoder Start) ###
    # 这一步只提取特征，不改变尺寸
```

```
    # 原注释：# x [3, T, 128,128] (注意：这里的3可能是指输入通道，通常输出通道会变大，如16)
    x = self.ConvBlock1(x)

    # 空间最大池化 (Spatial MaxPool)
    # 作用：也就是把图片变小，去噪，提取纹理。
    # 维度变化：128x128 -> 64x64 (时间 T 不变)
    x = self.MaxpoolSpa(x)  # 原注释：# x [16, T, 64,64]

    # ### 3. 中层特征提取 ###
    x = self.ConvBlock2(x)  # 原注释：# x [32, T, 64,64]

    # 这里特意用一个新变量 x_visual6464 接住结果
    # 目的：为了最后 return 的时候能把这个中间层特征输出来（可能用于辅助监督或可视化）
    x_visual6464 = self.ConvBlock3(x) # 原注释：# x [32, T, 64,64]

    # ### 4. 时空联合降采样 (Spatio-Temporal Downsampling) ###
    # 这一步很关键：不仅图片变小，时间也被压缩了！
    # 作用：融合相邻帧的信息，提取动态特征。
    # 维度变化：空间 64->32，时间 T->T/2
    # 原注释：# x [32, T/2, 32,32]    Temporal halve
    x = self.MaxpoolSpaTem(x_visual6464)

    # ### 5. 深层特征提取 ###
    x = self.ConvBlock4(x)  # 原注释：# x [64, T/2, 32,32]

    # 同样保留中间特征 x_visual3232
    x_visual3232 = self.ConvBlock5(x) # 原注释：# x [64, T/2, 32,32]

    # 再次进行时空压缩
    # 维度变化：空间 32->16，时间 T/2 -> T/4
    x = self.MaxpoolSpaTem(x_visual3232) # 原注释：# x [64, T/4, 16,16]

    x = self.ConvBlock6(x)  # 原注释：# x [64, T/4, 16,16]

    # 保留中间特征 x_visual1616
    x_visual1616 = self.ConvBlock7(x) # 原注释：# x [64, T/4, 16,16]

    # 最后一次单纯的空间池化 (时间不再压缩)
    # 维度变化：空间 16->8
```

```
    x = self.MaxpoolSpa(x_visual1616) # 原注释: # x [64, T/4, 8,8]

    # ### 6. 瓶颈层 (Bottleneck) ###
    # 此时特征图最小 (8x8)，但也最抽象，包含了全局信息
    x = self.ConvBlock8(x)  # 原注释: # x [64, T/4, 8, 8]
    x = self.ConvBlock9(x)  # 原注释: # x [64, T/4, 8, 8]

    # ### 7. 解码与时间恢复 (Upsampling) ###
    # 因为要输出 rPPG 信号是对应每一帧的，所以必须把被压缩的时间 T/4 还原回 T

    # 时间上采样: T/4 -> T/2
    x = self.upsample(x)  # 原注释: # x [64, T/2, 8, 8]

    # 时间上采样: T/2 -> T (现在时间长度和输入一样了)
    x = self.upsample2(x) # 原注释: # x [64, T, 8, 8]

    # ### 8. 信号生成头 (Prediction Head) ###
    # 原注释: # x [64, T, 1,1] --> groundtruth left and right - 7
    # 作用: 全局平均池化。把 8x8 的脸部特征图平均成 1x1 的点。
    # 物理含义: 忽略脸部具体位置 (哪里是鼻子哪里是脸颊)，只取整张脸的平均肤色变化特征。
    x = self.poolspa(x)

    # 降维: 把 64 个特征通道 压缩成 1 个通道 (也就是 rPPG 脉搏数值)
    x = self.ConvBlock10(x) # 原注释: # x [1, T, 1,1]

    # ### 9. 展平输出 ###
    # 将 [Batch, 1, T, 1, 1] 展平成 [Batch, T]
    # 这样每一行就是一个完整的脉搏波形
    rPPG = x.view(-1, length)

    # 返回最终预测信号 rPPG，以及前面保留的各个尺度的视觉特征
    return rPPG, x_visual, x_visual3232, x_visual1616
```

2. **02 深度学习基础理论学习** 以《深度学习花书》为基础，学习了项目相关理论知识，掌握了卷积神经网络的基本原理和工作机制
    - 相关理论知识有哪些？谈谈**卷积神经网络的基本原理和工作机制**。

- **相关理论知识** 我们学习了损失函数，学习了反向传播、优化器，了解了过拟合和欠拟合。
    - 损失函数：目标函数，将模型当前预测值和真实标签之间的差异映射为一个标量 Loss值。在我们的项目中体现的是在时域上的约束，用了负皮尔逊相关系数，因为rPPG任务更关心波形的 **趋势和形状** 是否一致。而心率的差异对比则通过 **MAE** 和 **RMSE** 对比
    - *前向传播*（forward）. 简单理解就是将上一层的输出作为下一层的输入，并计算下一层的输出，一直到运算到输出层为止。
    - 反向传播：计算损失函数对网络中每个参数（权重和偏置）的\梯度（Gradient），并调整参数使得我们的loss能变小
    - 优化器：根据反向传播计算出的**梯度**，来实际**更新**模型参数的算法。

      PhysFormer 的论文中明确指出使用了 Adam 优化器。Adam 是一种自适应优化器，它不仅考虑当前的梯度，还结合了历史梯度的一阶矩（均值）和二阶矩（方差）估计，通常比基础的随机梯度下降（SGD）收敛更快且更稳定。

      过拟合：模型在**训练集**上表现极好（Loss很低，精度很高），但在**测试集/验证集**上表现很差（Loss高，精度低）的现象。模型学习能力过强，不仅学到了数据中的普遍规律，还“死记硬背”了训练数据中的**噪声**（Noise）和**异常特征**（Outliers），导致泛化能力（Generalization）下降。体现：在 PhysFormer 论文中提到，如果只使用时域损失，容易导致过拟合 。

      欠拟合：模型在**训练集**和**测试集**上的表现都很差。模型的复杂度过低（参数太少或层数太浅），或者训练时间不足，导致模型**无法捕捉**数据中潜在的规律和特征。
- **CNN 基本原理：**
    - **核心机制** 主要是**卷积（Convolution）**、**池化（Pooling）和 激活函数（Activation）**。
    - **工作原理分为三个部分**，首先局部感知：卷积核像一个滑动窗口，在图像上滑动提取局部特征（如边缘、纹理），这也就是卷积部分。同一个卷积核在整张图上参数一样，大大减少了参数量。 浅层网络

提取边缘，深层网络提取语义（在rPPG中是皮肤颜色的微弱变化特征）

> 在滑动的每一个位置，卷积核的数值与输入数据对应位置的数值进行**点积运算**（对应位相乘再求和），最终生成一个新的特征图（Feature Map）。
>
> 池化是一种**下采样**（Downsampling）操作。它将输入特征图划分为若干个不重叠的矩形区域，然后对每个区域输出一个统计值；最大池化是取区域内最大值，平均池化取区域内平均值。
>
> 激活函数是附加在卷积操作之后的一个**非线性函数**，引入非线性，通常是逐元素（Element-wise）运算。 常见的有 **ReLU**（Rectified Linear Unit，修正线性单元，公式 $f(x) = max(0, x)$，即把负数变成0，正数保持不变）或 **Tanh** 等。

“**卷积层**负责通过点积运算从视频中**提取**时空特征（如肤色变化）； **激活函数**（如ReLU）负责引入**非线性**，让模型能拟合复杂的生理信号映射关系（否则会退化成了一个简单的线性滤波器）； **池化层**负责**下采样**，在保留关键特征的同时降低数据维度（去噪），减少计算量并防止过拟合。”

3. **03 论文阅读与学习** 系统学习两篇论文：PhysNet与PhysFormer，掌握了PhysNet利用3DCNN提取时空特征重建rPPG，以及PhysFormer引入时序差分Transformer增强长程感知与抗干扰能力的关键机制
    - **PhysNet** 的原理
    - **PhysFormer** 的原理
    - 为什么用 **3DCNN** 不用别的？比如2DCNN?
        - rPPG 信号本质上是**时间序列**上的颜色变化。2DCNN 只能处理单张图片（空间信息），它看不出“上一帧”和“下一帧”之间的联系，丢失了时间上下文 。但是3DCNN可以同时提取和处理空间和时间特征。
    - 时序差分 Transformer 和普通的 Transformer有什么区别？原理和功能上
        - 普通：核心是自注意力，基于“空间语义相似度”的匹配，通过原始图像特征学习相似度和关联性
        - 引入时序差分卷积后，输入不再是原始特征，而是**时序梯度**，开始学习变化特征。注意力权重与**信号的动态变化幅度**紧密相关，而非图像的静态外观。
        - **普通 Transformer：**

- 倾向于将**外观相似**的 Patch 聚类。例如，它可能会认为左眼和右眼是相关的，因为它们长得像。这对提取全脸平均脉搏波帮助有限，因为眼睛通常不包含强脉搏信号。

时序差分 Transformer：

- 倾向于将**变化规律一致**的 Patch 聚类。它会发现额头微血管的搏动相位与脸颊微血管的搏动相位是高度相关的（即 $Correlation(Grad_{forehead}, Grad_{cheek}) > 0$ ）。
- **效果：** 它能实现真正的**全局信号增强**，利用全脸所有皮肤区域的信号进行加权平均，从而大幅提高信噪比（SNR）。

- **PhysFormer** 和 **PhysNet** 都是对面部视频测量心率，他们的优缺点都有哪些?

PhysFormer Transformer架构 自注意力	**PhysNet 3D-CNN 卷积**
用transformer板块，权重是动态生成的；它不依赖固定的位置，而是依赖**信号的内容**。计算量大，显存占用高	结构简单，卷积核权重是固定的
模型会利用**全局一致性**，增强皮肤区域的权重，压制背景噪声。能捕捉长距离的时空联系	擅长提取局部时空特征 主要关注**局部邻域**（比如 $3 \times 3 \times 3$ 的像素块）。但在深层网络中，边缘信息容易丢失。
极易过拟合。如果没有足够的视频数据，或者没有精心设计的正则化策略（如 PhysFormer 论文中提到的 **Label Distribution Learning** 和 **Curriculum Learning**），它的效果往往不如 PhysNet。	计算量很小 PhysNet 更容易收敛，不容易过拟合。
	只能看到局部变化，难以捕捉长距离的时间依赖

维度	PhysNet (3D-CNN)	PhysFormer (Transformer)
核心运算	卷积 (Convolution)	自注意力 (Self-Attention)
处理逻辑	**静态权重**：不管输入怎么变，处理参数不变。	**动态权重**：根据输入内容动态调整关注点。
抗运动干扰	**弱**：依赖人脸检测和对齐，大动作易失效。	**强**：可通过 Attention 追踪移动的皮肤区域（Feature Alignment）。
信号一致性	**局部**：难以结合全脸信息互相验证。	**全局**：可同时利用额头、脸颊等多处信号互为参照去噪。
数据需求	**低**：自带归纳偏置，小样本也能训。	**高**：需要大量数据或强约束（Loss）来防止过拟合。
显存占用	线性增长 $O(N)$，比较省。	平方增长 $O(N^2)$，序列太长会爆显存。

4. **04 代码复现及对比** 使用RhythmFormer论文当中提供的rPPG-Toolbox进行训练和测试，并与论文给出的数据进行对比
    - 具体的训练和测试步骤是什么？配置是什么？和哪篇论文对比?

      在代码复现和对比实验环节，我们主要参考了 **Zou（邹）学者等学者** 发表的工作 **《RhythmFormer》**

      出于对**前沿基准**的遵循，我们的实验正是基于论文里提到的 rPPG-Toolbox 构建的。采用了论文中提供的标准训练与测试协议， 预处理为在第1帧将面部区域裁剪并调整大小为128 × 128，并在后续帧中固定该区域不再检测；配置为学习率设为了9e-3，训练30个epoch轮次；由于我们的显存有限，无法满足论文配置里提到的batchsize批大小 = 4，我们采用了 梯度累积（Gradient Accumulation） 策略进行适配，将物理批大小调整为 = 2，但是在训练代码中进行梯度累积、设置累积步数为2—也就是每两次前向传播后进行一次反向传播更新，从

而在数学上实现了等效batchsize=4的效果，其他没有提到的配置则为toolbox默认配置，确保了与原论文训练条件的严格对齐；**我们理解代码和论文后，**修改了工具箱里的训练代码和加载数据代码以适配我们自身情况，训练并测试，将我们复现的 PhysNet 和 PhysFormer 模型效果，与论文对两个模型的复现后报告的数据进行了严谨的对齐和对比。

训练原理

1. 首先，数据通过模型前向传播得到预测信号，我们利用**损失函数**（如负皮尔逊相关）来量化它与真实脉搏波的差异。
2. 接着，通过**反向传播**算法，利用链式法则计算出损失函数对每个模型参数的梯度。
3. 然后，**Adam 优化器**根据这些梯度，自适应地更新模型参数，使 Loss 逐步降低。
4. 在整个过程中，我们极力避免两个极端：一是**欠拟合**，即模型太简单学不到特征（所以我们用了深层的 3DCNN/Transformer）；二是**过拟合**，即模型学到了噪声导致泛化差（所以 PhysFormer 引入了课程学习等正则化策略来约束模型）。”

实验解释

> 模型 on 数据集 的数据解释

MAE，平均绝对误差，预测心率和真实心率差值的绝对值的平均

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

RMSE，均方根误差，对大误差更敏感

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Loss, 负皮尔逊相关系数

$$r = \frac{\sum_{i=1}^{N}(y_i-\bar{y})(\hat{y}_i-\bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i-\bar{y})^2}\cdot\sqrt{\sum_{i=1}^{N}(\hat{y}_i-\bar{\hat{y}})^2}}$$
$$Loss = 1 - r$$

- **PhysNet**
  - **PURE**

    模型在正常心率范围内（40-100bpm），精度很高，在少数极端心率范围内出现了倍频错误，导致误差较大。

    - Loss 图
    - 学习率图

        - BA（2个）图
        - MAE, loss, RMSE 含义与对比
    - **UBFC**

      模型效果很好，大部分心率情况下都非常接近实际心率信号，部分离散点可能是视频中出现了比较大的运动情况，导致模型短时间内丢失了信号

        - Loss 图
        - 学习率图
        - BA（2个）图
        - MAE, loss, RMSE 含义与对比
    - **MMPD**

        - Loss 图
        - 学习率图
        - BA（2个）图
        - MAE, loss, RMSE 含义与对比
- **PhysFormer**
    - **PURE**

      40-80bpm的精度很高，但是部分数据点可能由于分频错误或者运动噪声影响，误差较大。

        - Loss 图
        - 学习率图
        - BA（2个）图
        - MAE, loss, RMSE 含义与对比
    - **UBFC**

      数据精度很高，可能与UBFC测试集的人物运动较少和光照情况良好有关。

        - Loss 图
        - 学习率图
        - BA（2个）图
        - MAE, loss, RMSE 含义与对比
    - **MMPD**

        - Loss 图
        - 学习率图
        - BA（2个）图

        - MAE, loss, RMSE 含义与对比

未来改进方向分析

1. **现有基于Transformer的rPPG方法在受控环境下表现优异，但在面向非受控自然场景时仍存在鲁棒性不足。**
    - 如何体现鲁棒性不足?
        - 1.现有模型缺乏对复杂光照环境的自适应感知能力2.受试者大幅度头部运动及面部表情变化
        -
2. 引入**通道注意力模块**自适应重校准特征权重，实现生理信号与光照噪声的有效解耦.
    - 这个模块的原理是什么?
    - 怎么实现解耦？解耦的意思是什么?
    - 对鲁棒性有什么帮助?
    - 最经典的实现是 **SE-Block (Squeeze-and-Excitation)**。
        - **输入：** 假设你的特征图有 C 个通道（比如 RGB 3个通道，或者中间层 64 个特征通道）。
        - **Squeeze（压缩）：** 先把每个通道的空间信息压缩成一个数（全局平均池化），得到一个 $1 \times 1 \times C$ 的向量。这代表了每个通道的“全局重要性”。
        - **Excitation（激励）：** 通过两个全连接层（FC），学习出一组**权重**（Weight），范围是 0 到 1。
        - **Reweight（重校准）：** 用这组权重去乘回原来的特征图。

      直观理解：

      模型仿佛在这个模块里问自己：“现在的环境光是偏红的，红色通道里全是噪声，我该信谁？”

        - 它自动学出的权重可能是：`Red: 0.1` (不可信), `Green: 0.9` (脉搏主要在这里), `Blue: 0.2` (辅助)。

      ### 2. 什么是“解耦 (Decoupling)”?

      在 rPPG 语境下，**解耦 = 数学上的信号分离**。

      原始信号 S_{input} 是混合的：

      $$S_{input} = S_{\text{生理脉搏}} + S_{\text{光照变化}} + S_{\text{运动噪声}}$$

        - **未解耦前：** 所有的噪声都混在一起，模型很难分清哪个波动是心跳，哪个是灯光闪烁。
        - **解耦后：** 通道注意力机制发现，$S_{\text{光照变化}}$ 主要集中在某些特定的特征通道上（比如那些对亮度极其敏感的通道）。于

是，它给这些“光照敏感通道”赋予极低的权重（乘以0.01），相当于把光照噪声**剥离（解耦）**出去了，只留下了干净的生理信号通道。

3. 采用可变形卷积替代部分标准卷积，以自适应捕捉面部细微几何形变，解决表情变化导致的特征非对齐难题。
    - 可变形卷积和标准卷积的区别是什么?
    - 为什么可以解决这个问题?
    - **可变形卷积 (Deformable Conv)**：
        - **形状：** 网格是**不规则**的，可以扭曲。
        - **采样点：** 在采样之前，网络先学习一个**偏移量 (Offset)** $(\Delta x, \Delta y)$。
        - **对齐 (Alignment) 的概念：** 在时间序列中，第 t 帧的皮肤点 P，在第 $t+1$ 帧可能移到了 P'。

          自适应捕捉： 当受试者大笑时，可变形卷积的采样点会自动“拉伸”，紧紧咬住那块含有脉搏信号的皮肤区域，而不会误读到嘴巴内部（没有脉搏）的像素。

          结果： 无论人脸怎么扭曲，提取到的特征始终来自于**同一块皮肤区域**，保证了时序信号的连续性和纯净度。
4. 我们将开展多模态数据采集工作，**同步采集毫米波雷达、热红外图像、RGB图像和指尖脉搏波等多源数据**，在此基础上构建全新的多模态rPPG数据集。
    - 具体的采集方法，采集打算?
    - 构建方法?
    - 采集这些4种数据和之前的训练数据有什么区别?

可能的其他问题

- 你在这个复现过程中遇到的最大困难是什么?